Student Name : Sankar Samiksha

Group : SS5

Date : 5 April 2022

LAB 4: ANALZING NETWORK DATA LOG

You are provided with the data file, in .csv format, in the working directory. Write the program to extract the following informations.

EXERCISE 4A: TOP TALKERS AND LISTENERS

One of the most commonly used function in analyzing data log is finding out the IP address of the hosts that send out large amount of packet and hosts that receive large number of packets, usually know as TOP TALKERS and LISTENERS. Based on the IP address we can obtained the organization who owns the IP address.

List the TOP 5 TALKERS

Rank	IP address	# of packets	Organisation
1	193.62.192.8	3041	European Bioinformatics, England
2	155.69.160.32	2975	NTU, Singapore
3	130.14.250.11	2604	National Library of Medicine, United States
4	14.139.196.58	2452	Indian Institute of Technology (IIT) Guwahati, India
5	140.112.8.139	2056	Taiwan Academic Network, Taiwan

TOP 5 LISTENERS

Rank	IP address	# of packets	Organisation
1	103.37.198.100	3841	A*STAR, Singapore
2	137.132.228.15	3715	NUS, Singapore
3	202.21.159.244	2446	Republic Polytechnic, Singapore
4	192.101.107.153	2368	Pacific Northwest National Laboratory, United States
5	103.21.126.2	2056	Powai, India

EXERCISE 4B: TRANSPORT PROTOCOL

Using the IP protocol type attribute, determine the percentage of TCP and UDP protocol

	Header value	Transport layer protocol	# of packets
1	6	TCP	56064
2	17	UDP	9462
3	50	ESP	1698

Total Number of Packets in the entire CSV: 69370
Percentage of UDP: 13.639901974917112
Percentage of TCP: 80.81879775118928

EXERCISE 4C: APPLICATIONS PROTOCOL

Using the Destination IP port number determine the most frequently used application protocol. (For finding the service given the port number https://www.adminsub.net/tcp-udp-port-finder/)

Rank	Destination IP port number	# of packets	Service
1	443	13423	https
2	80	2647	http
3	52866	2068	Dynamic and/or private ports (UDP) Xsan. Xsan Filesystem Access(TCP)
4	45512	1356	Unassigned
5	56152	1341	Dynamic and/or private ports Xsan. Xsan Filesystem Access(TCP)

EXERCISE 4D: TRAFFIC

The traffic intensity is an important parameter that a network engineer needs to monitor closely to determine if there is congestion. You would use the IP packet size to calculate the estimated total traffic over the monitored period of 15 seconds. (Assume the sampling rate is 1 in 2048)

IP packet size corresponds to the IP_Size column name

Total Traffic(MB)	132664.98 MB (to 2 d.p)

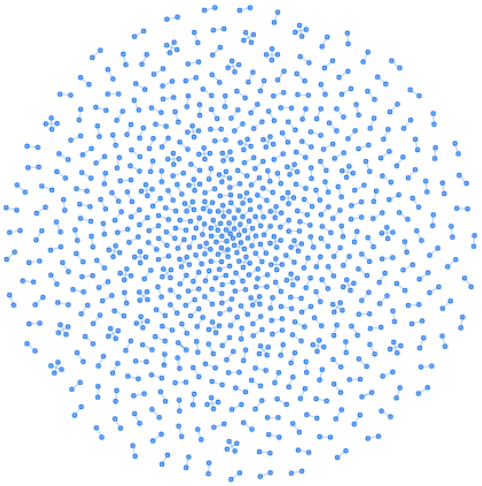
EXERCISE 4E: ADDITIONAL ANALYSIS

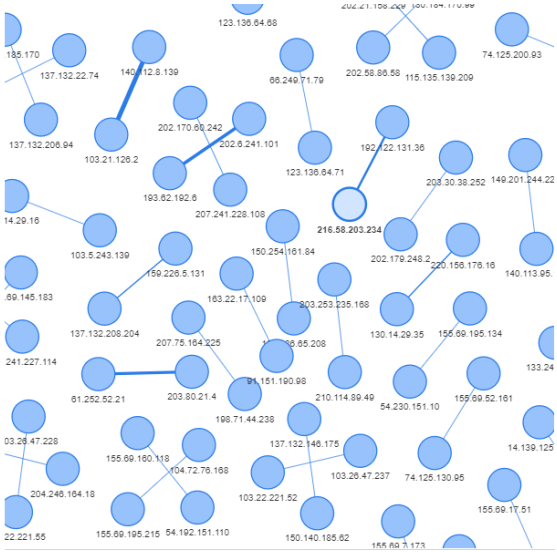
Please append ONE page to provide additional analysis of the data and the insight it provides.
Examples include:
Top 5 communication pairs;
Visualization of communications between different IP hosts;
etc.
Please limit your results within one page (and any additional results that fall beyond one page limit will not be assessed).

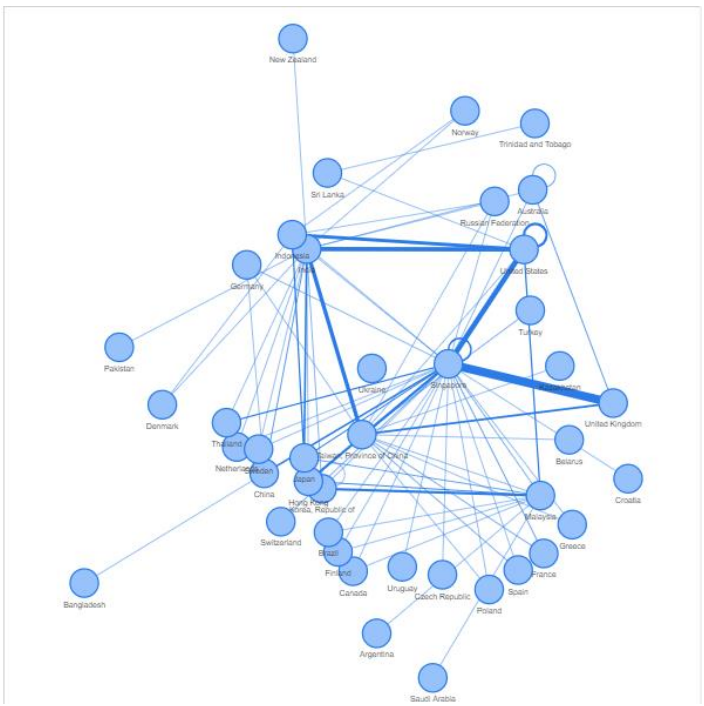
The top 5 communication pairs

IP Address 1	IP Address 2	Count
137.132.228.15	193.62.192.8	4951
130.14.250.11	103.37.198.100	2842
14.139.196.58	192.101.107.153	2368
103.21.126.2	140.112.8.139	2056
167.205.52.8	140.90.101.61	1752

Using the table of communication pairs the network graph below was plotted. The graph is very sparse and most nodes are only connected to one other node. The image in the right is a zoomed in version. The thickness of the line indicates the count or the amount the two ip addresses have interacted with each other.







Later, the ip addresses were grouped accordingly the country of origin. Below shows the graph:
A hollow circle indicates that the two ip addresses interacting with each other come from the same country. Similar to earlier, the thicker the line, the more the interaction. Here we can see that Singapore interacts most with United Kingdom followed by United States.

EXERCISE 4F: SOFTWARE CODE

Please also submit your code to the NTULearn lab site.

CZ3006 Lab 4

Aim: Doing basic analysis of data log

In [1]:
```
# Basic Libraries
import numpy as np
import pandas as pd
```

Creating the dataframe using pandas library

1. list the column names required
2. import the csv adding in the column names

In [2]:
```
colName = ['Type', 'sflow_agent_address', 'inputPort', 'outputPort', 'src_Mac', 'dst
SFlow = pd.read_csv('./SFlow_Data_lab4.csv', header = None, names=colName, index_col
SFlow.head()
```

Out[2]:

	Type	sflow_agent_address	inputPort	outputPort	src_Mac	dst_Mac	ethernet_type	i
0	FLOW	aa.aa.aa.aa	137	200	d404ff55fd4d	80711fc76001	0x0800	
1	FLOW	aa.aa.aa.aa	129	193	609c9f851b00	0031466b23cf	0x0800	
2	FLOW	aa.aa.aa.aa	137	200	d404ff55fd4d	80711fc76001	0x0800	
3	FLOW	aa.aa.aa.aa	129	135	609c9f851b00	002688cd5fc7	0x0800	
4	FLOW	aa.aa.aa.aa	130	199	00239cd087c1	544b8cf9a7df	0x0800	

EXERCISE 4A: TOP TALKERS AND LISTENERS

One of the most commonly used function in analyzing data log is finding out the IP address of the hosts that send out large amount of packet and hosts that receive large number of packets, usually know as TOP TALKERS and LISTENERS. Based on the IP address we can obtained the organization who owns the IP address.

The organizations are found using the link https://whatismyipaddress.com/

In [3]:
```
#aim is to find the top 5 talkers using IP address
table1 = SFlow.loc[SFlow['Type'] == 'FLOW']
table1 = pd.DataFrame(table1['src_ip'])
Top5Talkers = table1.value_counts()
Top5Talkers.head()
```

Out[3]:
```
src_ip
193.62.192.8     3041
155.69.160.32    2975
130.14.250.11    2604
14.139.196.58    2452
140.112.8.139    2056
dtype: int64
```

In [4]:
```
#aim is to find the top 5 listeners using IP address
table1 = SFlow.loc[SFlow['Type'] == 'FLOW']
table1 = pd.DataFrame(table1['dst_ip'])
Top5Listeners= table1.value_counts()
Top5Listeners.head()
```

```
dst_ip
```

Out[4]:
```
103.37.198.100     3841
137.132.228.15     3715
202.21.159.244     2446
192.101.107.153    2368
103.21.126.2       2056
dtype: int64
```

EXERCISE 4B: TRANSPORT PROTOCOL

Using the IP protocol type attribute, determine the percentage of TCP and UDP protocol

In [5]:
```
table1 = pd.DataFrame(SFlow['IP_protocol']) #take out IP_protocol only
IPprotocol = table1.value_counts() #count the number of unique occurrences
IPprotocol = IPprotocol.to_frame().reset_index() #reset the index
IPprotocol.columns = ["IP_Protocol", "packets"] #rename the column
print(IPprotocol) #print the table
totalpackets = IPprotocol['packets'].sum()
print()
# printing all the statistics
print("Total number of packets sent : ", totalpackets)
print("Percentage of UDP", 10*" ", ": ", np.array(IPprotocol.loc[IPprotocol['IP_Prot
print("Percentage of TCP", 10*" ", ": ", np.array(IPprotocol.loc[IPprotocol['IP_Prot
```

```
   IP_Protocol  packets
0            6    56064
1           17     9462
2           50     1698
3            0     1261
4           47      657
5           41      104
6            1       74
7          381       45
8           58        4
9          103        1

Total number of packets sent :  69370
Percentage of UDP            :  13.639901974917112
Percentage of TCP            :  80.81879775118928
```

EXERCISE 4C: APPLICATIONS PROTOCOL

Using the Destination IP port number determine the most frequently used application protocol.

(For finding the service given the port number https://www.adminsub.net/tcp-udp-port-finder/)

In [6]:
```
table1 = pd.DataFrame(SFlow['udp_dst_port/tcp_dst_port'])
Top5Apps = table1.value_counts()
Top5Apps = Top5Apps.to_frame().reset_index()
Top5Apps.columns = ["udp_dst_port/tcp_dst_port", "packets"]
print(Top5Apps.head(5))
totalpackets = Top5Apps['packets'].sum()
print()
print("Total number of packets sent : ", totalpackets)
```

```
   udp_dst_port/tcp_dst_port  packets
0                        443    13423
1                         80     2647
2                      52866     2068
3                      45512     1356
4                      56152     1341

Total number of packets sent :  69370
```

EXERCISE 4D: TRAFFIC

The traffic intensity is an important parameter that a network engineer needs to monitor closely to determine if there is congestion. You would use the IP packet size to calculate the estimated total traffic over the monitored period of 15 seconds. (Assume the sampling rate is 1 in 2048)

IP packet size corresponds to the IP_Size column name

In [7]:
```
table1 = pd.DataFrame(SFlow['IP_Size'])
totalbytes = table1.sum()
totalMB = totalbytes / 1000 /1000 * 2048
print("Total Traffic (MB) : ", np.array(totalMB)[0])
```

```
Total Traffic (MB) :  132664.979456
```

EXERCISE 4E: ADDITIONAL ANALYSIS

Please append ONE page to provide additional analysis of the data and the insight it provides. Examples include: Top 5 communication pairs; Visualization of communications between different IP hosts; etc. Please limit your results within one page (and any additional results that fall beyond one page limit will not be assessed).

Finding the top 5 communication pair

We have to take into account the fact that src --> dst and dst --> src both have to be counted as a communication pair. For example, two such entries:

src | dst 137.132.228.15 193.62.192.8
193.62.192.8 137.132.228.15

means between the IP addresses `137.132.228.15` and `193.62.192.8` there have been **2** communications.

In [8]:
```
#make this neater
table1 = pd.DataFrame(SFlow[['src_ip', 'dst_ip']])
table2 = pd.DataFrame(SFlow[['src_ip', 'dst_ip']])
table1.rename(columns = {'dst_ip': 'IP1', 'src_ip': 'IP2'}, inplace = True)
table2.rename(columns = {'dst_ip': 'IP2', 'src_ip': 'IP1'}, inplace = True)
finaltable = pd.concat([table1, table2])
commpair = (finaltable.groupby(['IP1', 'IP2'])).size().sort_values(ascending=False).
commpair['index'] = commpair.index
df_1 = commpair[['IP1', 'index', 'count']]
df_2 = commpair[['IP2', 'index', 'count']]
df_1.columns = ['IP', 'index', 'count']
df_2.columns = ['IP', 'index', 'count']
df_1['source'] = 1
df_2['source'] = 2
df = pd.concat([df_1, df_2])
out = df.sort_values(['index']).drop_duplicates(['IP'], keep='first')
df_1_out = out[out['source'] == 1][['IP', 'count', 'index']]
df_2_out = out[out['source'] == 2][['IP', 'count', 'index']]
final = df_1_out.merge(df_2_out, on='index', suffixes=('_1', '_2')).drop('index', ax
final = final.drop(columns = ['count_1'])
final.rename(columns = {'count_2': 'count'}, inplace=True)
#view the top 5 commmunication pairs
print(final.head(5))
```

```
              IP_1             IP_2  count
0  137.132.228.15     193.62.192.8   4951
1    130.14.250.11   103.37.198.100   2842
2    14.139.196.58  192.101.107.153   2368
```

```
3    103.21.126.2    140.112.8.139   2056
4    167.205.52.8    140.90.101.61   1752
```

Visualizing these communication pairs (By IP Address)

There are two parts to this:

1. Visualising based on the IP address of the sending and receiving hosts
2. Based on the location where these IP addresses are originating from

In [9]:
```
commpair = final.head(500)
```

In [10]:
```
import sys
!{sys.executable} -m pip install pyvis
```

```
Requirement already satisfied: pyvis in c:\users\samik\anaconda3\lib\site-packages
(0.1.9)
Requirement already satisfied: networkx>=1.11 in c:\users\samik\anaconda3\lib\site-p
ackages (from pyvis) (2.5)
Requirement already satisfied: jsonpickle>=1.4.1 in c:\users\samik\anaconda3\lib\sit
e-packages (from pyvis) (2.1.0)
Requirement already satisfied: ipython>=5.3.0 in c:\users\samik\anaconda3\lib\site-p
ackages (from pyvis) (7.19.0)
Requirement already satisfied: jinja2>=2.9.6 in c:\users\samik\anaconda3\lib\site-pa
ckages (from pyvis) (2.11.2)
Requirement already satisfied: decorator>=4.3.0 in c:\users\samik\anaconda3\lib\site
-packages (from networkx>=1.11->pyvis) (4.4.2)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in c:\us
ers\samik\anaconda3\lib\site-packages (from ipython>=5.3.0->pyvis) (3.0.8)
Requirement already satisfied: pygments in c:\users\samik\anaconda3\lib\site-package
s (from ipython>=5.3.0->pyvis) (2.7.2)
Requirement already satisfied: colorama; sys_platform == "win32" in c:\users\samik\a
naconda3\lib\site-packages (from ipython>=5.3.0->pyvis) (0.4.4)
Requirement already satisfied: backcall in c:\users\samik\anaconda3\lib\site-package
s (from ipython>=5.3.0->pyvis) (0.2.0)
Requirement already satisfied: pickleshare in c:\users\samik\anaconda3\lib\site-pack
ages (from ipython>=5.3.0->pyvis) (0.7.5)
Requirement already satisfied: jedi>=0.10 in c:\users\samik\anaconda3\lib\site-packa
ges (from ipython>=5.3.0->pyvis) (0.17.1)
Requirement already satisfied: traitlets>=4.2 in c:\users\samik\anaconda3\lib\site-p
ackages (from ipython>=5.3.0->pyvis) (5.0.5)
Requirement already satisfied: setuptools>=18.5 in c:\users\samik\anaconda3\lib\site
-packages (from ipython>=5.3.0->pyvis) (50.3.1.post20201107)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\samik\anaconda3\lib\site
-packages (from jinja2>=2.9.6->pyvis) (1.1.1)
Requirement already satisfied: wcwidth in c:\users\samik\anaconda3\lib\site-packages
(from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=5.3.0->pyvis) (0.2.5)
Requirement already satisfied: parso<0.8.0,>=0.7.0 in c:\users\samik\anaconda3\lib\s
ite-packages (from jedi>=0.10->ipython>=5.3.0->pyvis) (0.7.0)
Requirement already satisfied: ipython-genutils in c:\users\samik\anaconda3\lib\site
-packages (from traitlets>=4.2->ipython>=5.3.0->pyvis) (0.2.0)
```

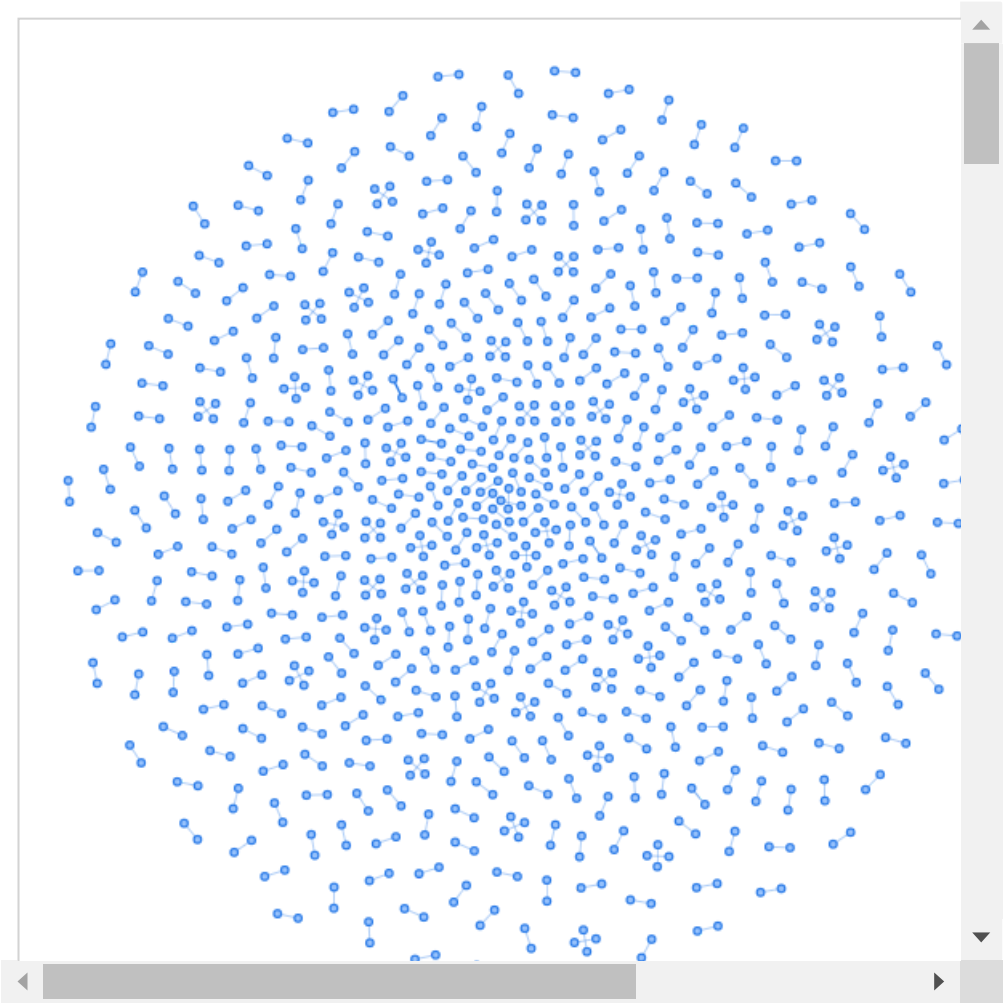
In [11]:
```
from pyvis.network import Network
```

In [12]:
```
commgraph = Network(notebook = True)
#firstly must have a unique list of ip hosts
iphost = pd.DataFrame(commpair[['IP_1']])
iphost2 = pd.DataFrame(commpair[['IP_2']])
iphost.rename(columns = {'IP_1': 'IP'}, inplace = True)
iphost2.rename(columns = {'IP_2': 'IP'}, inplace = True)
finaltable = pd.concat([iphost, iphost2])
finaltable = finaltable.drop_duplicates('IP').reset_index(drop=True)
#print(finaltable)
for i in finaltable.index:
    commgraph.add_node(str(finaltable.loc[i, "IP"]))
```

```
#commgraph.show('nodes.html')
```

In [13]:
```
# the list of edges is stored in commpair
for i in commpair.index:
    commgraph.add_edge(str(commpair.loc[i, "IP_1"]), str(commpair.loc[i, "IP_2"]), v

commgraph.show_buttons(filter_=True)
commgraph.show('nodes.html')
```

Out[13]:



Visualizing these communication pairs (By Location)

There are two parts to this:

1. Visualising based on the IP address of the sending and receiving hosts
2. Based on the location where these IP addresses are originating from

In [14]:
```
!{sys.executable} -m pip install IP2Location
import os
import IP2Location

ip = '137.132.228.15'

database = IP2Location.IP2Location(os.path.join("IP-COUNTRY.BIN"))

rec = database.get_all(ip)
print(rec.country_long)
print(rec.country_short)
```

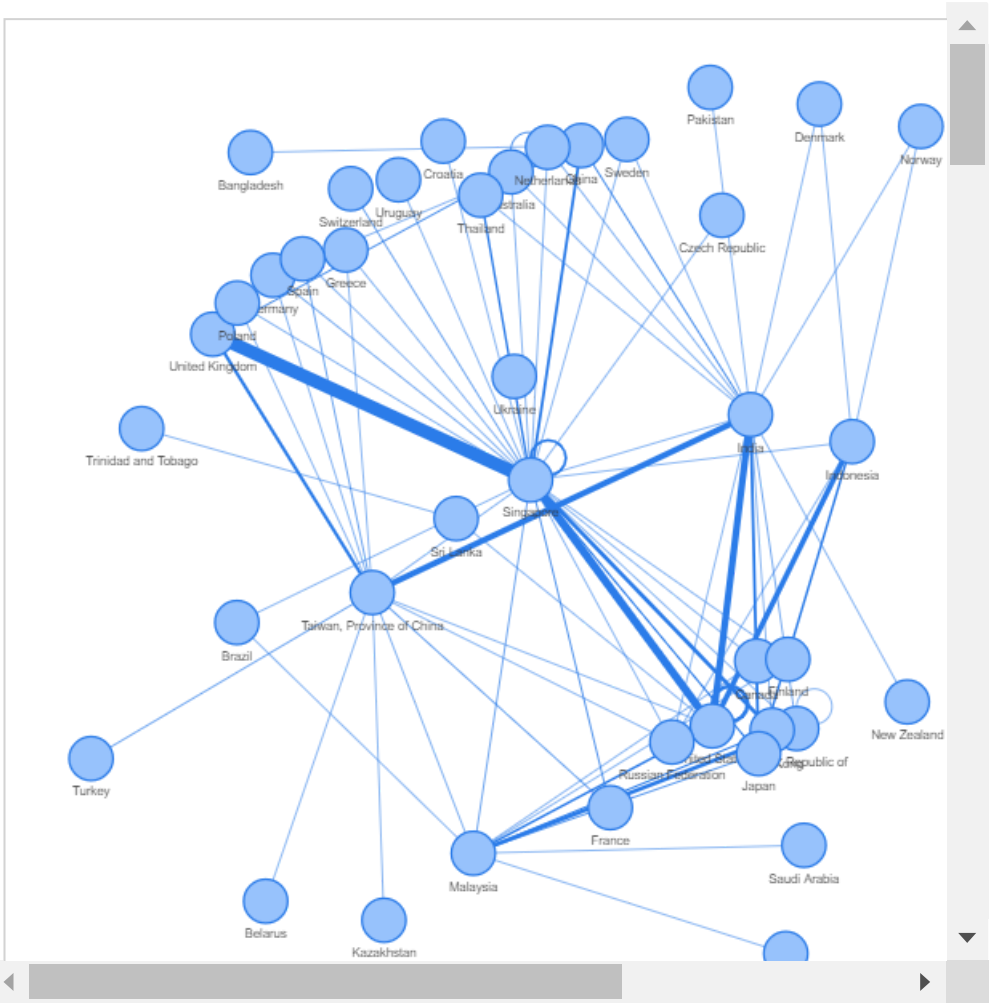
```
Requirement already satisfied: IP2Location in c:\users\samik\anaconda3\lib\site-pack
ages (8.7.2)
```

```
Singapore
SG
```

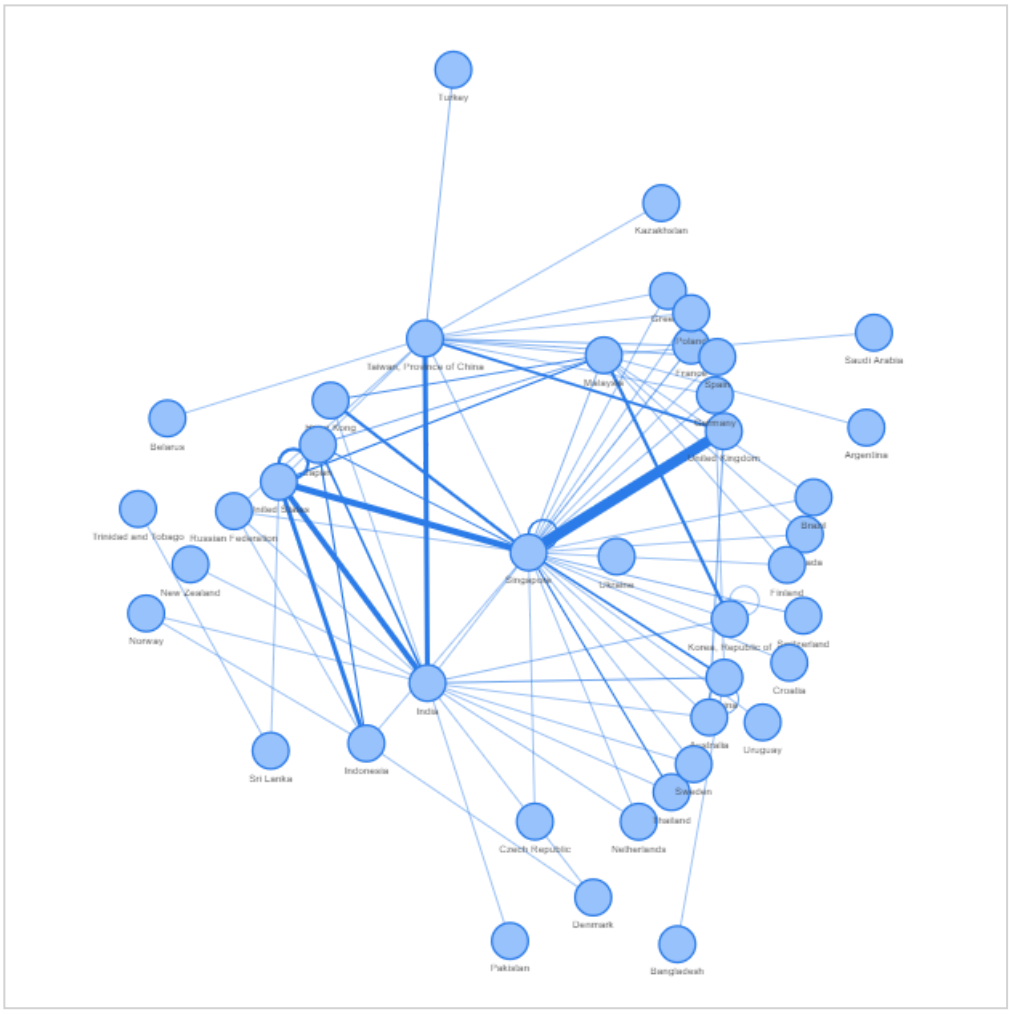
In [15]:
```
locgraph = Network(notebook = True)
for i in finaltable.index:
    rec = database.get_all(str(finaltable.loc[i, "IP"]))
    if (rec.country_long != 'INVALID IP ADDRESS' and rec.country_long != 'IPV6 ADDRE
        locgraph.add_node(str(rec.country_long))
```

In [16]:
```
# the list of edges is stored in commpair

for i in commpair.index:
    rec1 = database.get_all(str(commpair.loc[i, "IP_1"]))
    city1 = str(rec1.country_long)
    rec2 = database.get_all(str(commpair.loc[i, "IP_2"]))
    city2 = str(rec2.country_long)

    if (city1 != 'INVALID IP ADDRESS' and city1 != 'IPV6 ADDRESS MISSING IN IPV4 BIN
        if (city2 != 'INVALID IP ADDRESS' and city2 != 'IPV6 ADDRESS MISSING IN IPV4
            #print(city1, city2)
            locgraph.add_edge(city1, city2, value = int(commpair.loc[i, "count"]))


locgraph.repulsion(node_distance=70, spring_length=250)
locgraph.show_buttons(filter_=True)
locgraph.show('locgraph.html')
```

Out[16]:



In [ ]:



nodes

borderWidth: 1

borderWidthSelected: 2

color:

border:

background:

highlight:

border:

background:

hover:

border:

background:

fixed:

x:

y:

font:

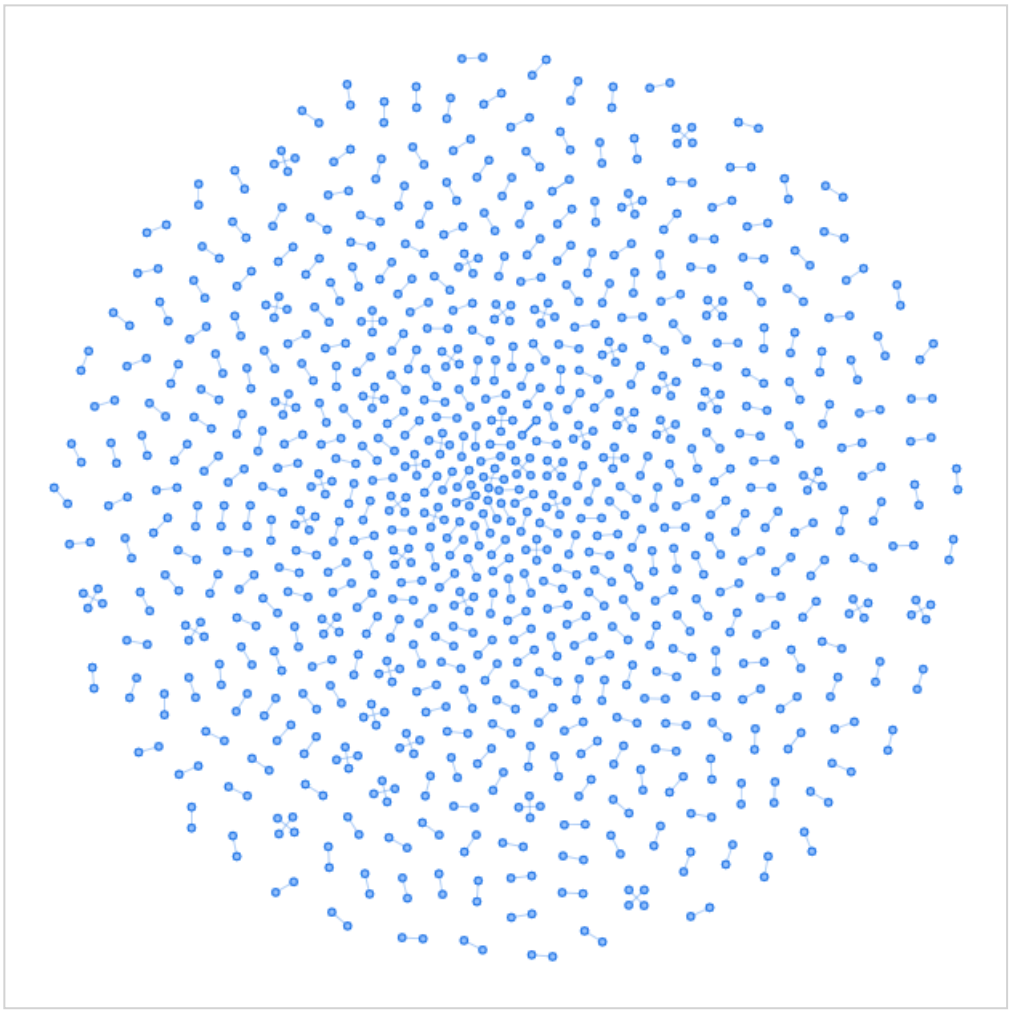
color:

size: 14

face: arial

background:

strokeWidth: 0



nodes

borderWidth: 1

borderWidthSelected: 2

color:

border:

background:

highlight:

border:

background:

hover:

border:

background:

fixed:

x:

y:

font:

color:

size: 14

face: arial

background:

strokeWidth: 0